

Computer Olympic 2022

พอยน์เตอร์ (Pointer)

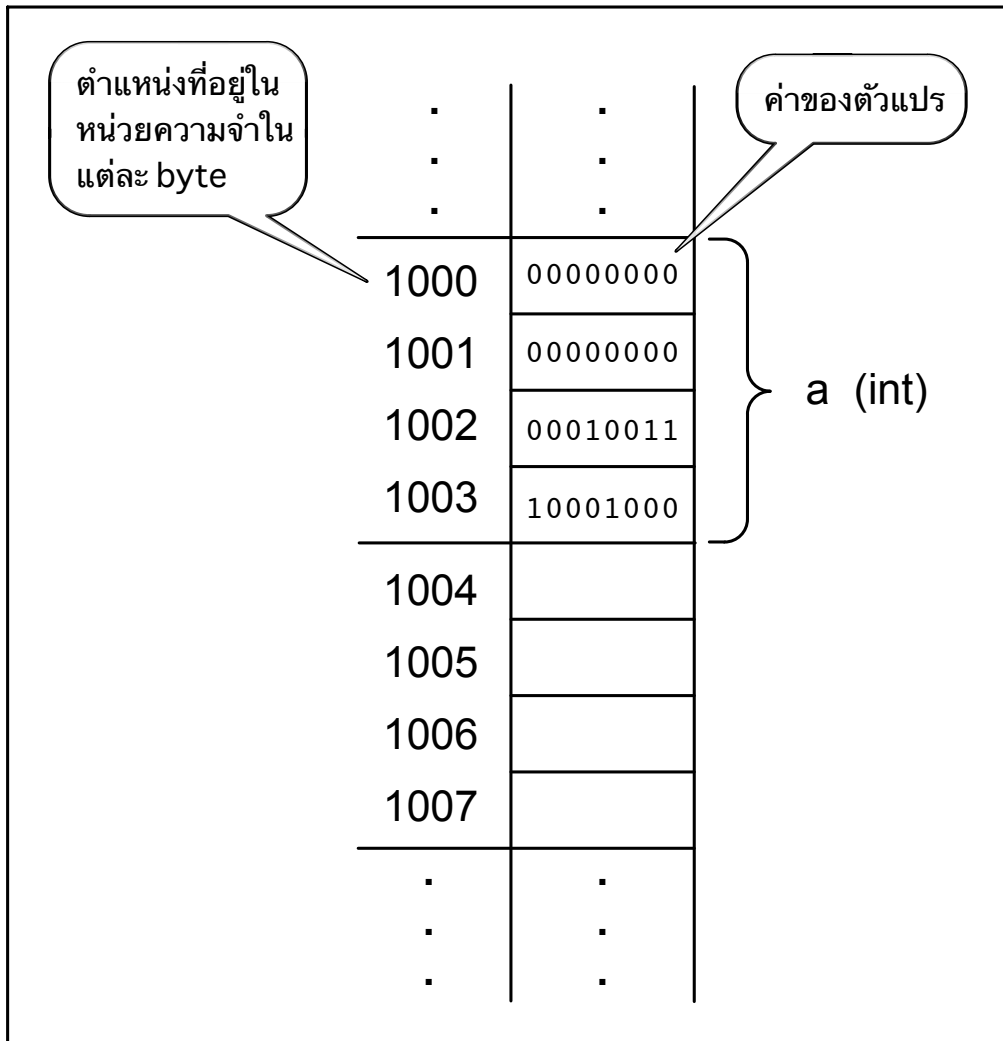
Outline

- การเก็บข้อมูลในหน่วยความจำ
- การใช้พอยน์เตอร์
- พอยน์เตอร์ชี้ไปยังพอยน์เตอร์
- พอยน์เตอร์กับอาร์เรย์
- Dynamic array

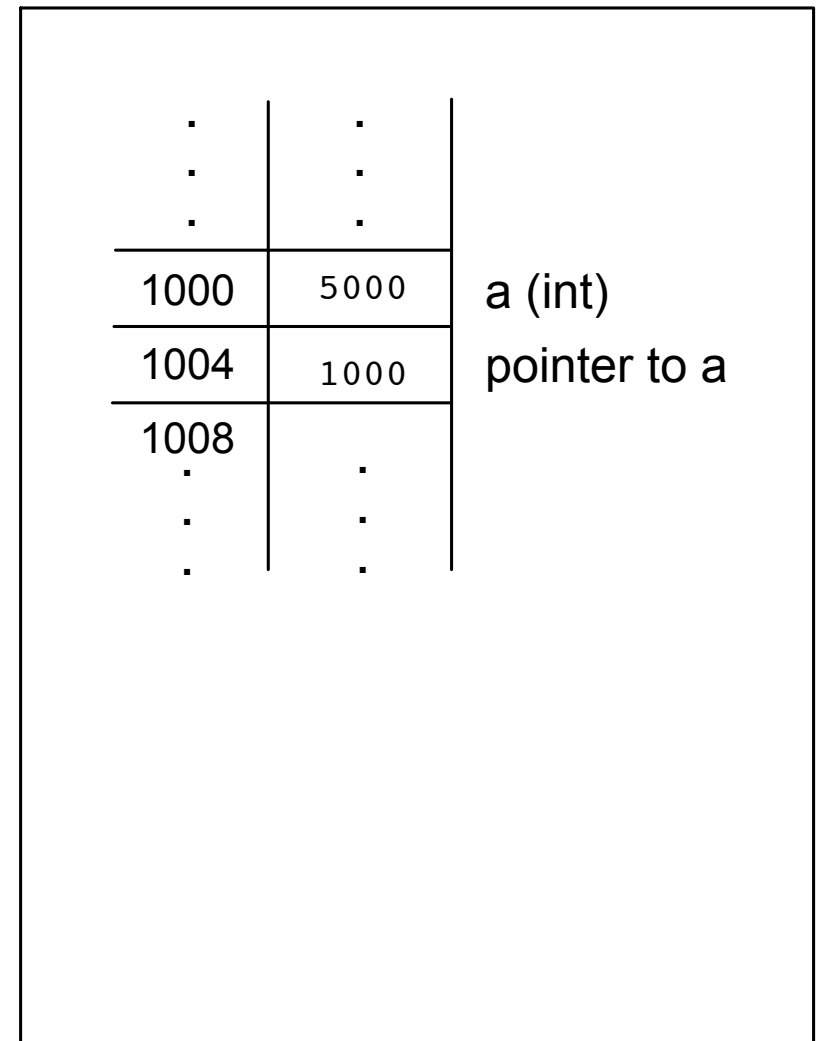
พอยน์เตอร์ (Pointer)

- ตัวแปรพอยน์เตอร์จะ เก็บตำแหน่งที่อยู่ในหน่วยความจำ (memory address) ของตัวแปรแทนที่จะเก็บค่าข้อมูลเหมือนตัวแปรชนิดอื่น
- ตัวแปรพอยน์เตอร์จึงมองดูเหมือนกับตัวชี้หรือพอยน์เตอร์ชี้ไปที่ตำแหน่งที่อยู่

ทบทวน เรื่องหน่วยความจำ



a) การเก็บค่าตัวแปรชนิด int ซึ่งใช้เนื้อที่ 4 byte ในหน่วยความจำ



b) การเก็บค่าตัวแปรชนิด int และพอยน์เตอร์ ซึ่งทั้งสองใช้เนื้อที่ 4 byte ในหน่วยความจำ

การตรวจสอบขนาดของชนิดข้อมูล

● ใช้ฟังก์ชัน `sizeof`

```
#include <stdio.h>
int main()
{
    printf("Size of a short is %d bytes.\n", sizeof(short));
    printf("Size of a int is %d bytes.\n", sizeof(int));
    printf("Size of a long is %d bytes.\n", sizeof(long));
    return 0;
}
```

Size of a short is 2 bytes.
Size of a int is 4 bytes.
Size of a long is 8 bytes.


การใช้พอยน์เตอร์

- การประกาศตัวแปรพอยน์เตอร์ต้องมีเครื่องหมาย * (asterisk) หน้าชื่อตัวแปร


```
type* name;
```


การใช้พอยน์เตอร์


- เปรียบเทียบการประกาศตัวแปรชนิดข้อมูลพื้นฐาน และการประกาศตัวแปรพอยน์เตอร์


int a; 

float b; 

char c; 

int* x;  <int>

float* y;  <float>

char* z;  <char>

การกำหนดค่าให้พอยน์เตอร์

- การกำหนดค่าให้พอยน์เตอร์ ใช้เครื่องหมาย & (ampersand) แทนตำแหน่งที่อยู่ในหน่วยความจำของตัวแปร
- ตัวอย่างเช่น

```
ptr = &x;
```


การกำหนดค่าให้พอยน์เตอร์

การประกาศตัวแปร

```
int* ptr;
```

การกำหนดค่า

```
int x;
```

```
x = 100;
```

```
ptr = &x;
```

ตำแหน่งที่อยู่

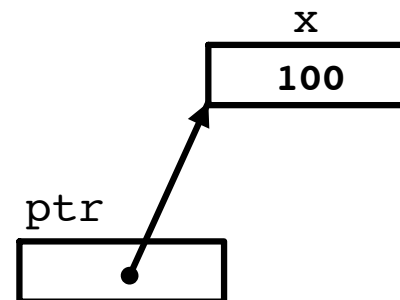
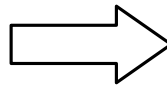
x
345674

ptr
345678

a) การเก็บค่าในหน่วยความจำหลังจากประกาศตัวแปร x และ ptr

x
345674 100

ptr
345678 345674



b) การเก็บค่าในหน่วยความจำหลังจากการกำหนดค่าตัวแปรหรือสามารถแสดงได้ในลักษณะของพอยน์เตอร์ที่เป็นตัวชี้

การเรียกใช้พอยน์เตอร์

- ใช้เครื่องหมาย * นำหน้าพอยน์เตอร์

*ptr หมายถึง ค่าในหน่วยความจำของตำแหน่งที่ ptr ชี้อยู่

การเรียกใช้พอยน์เตอร์

```
int* ptr;
```

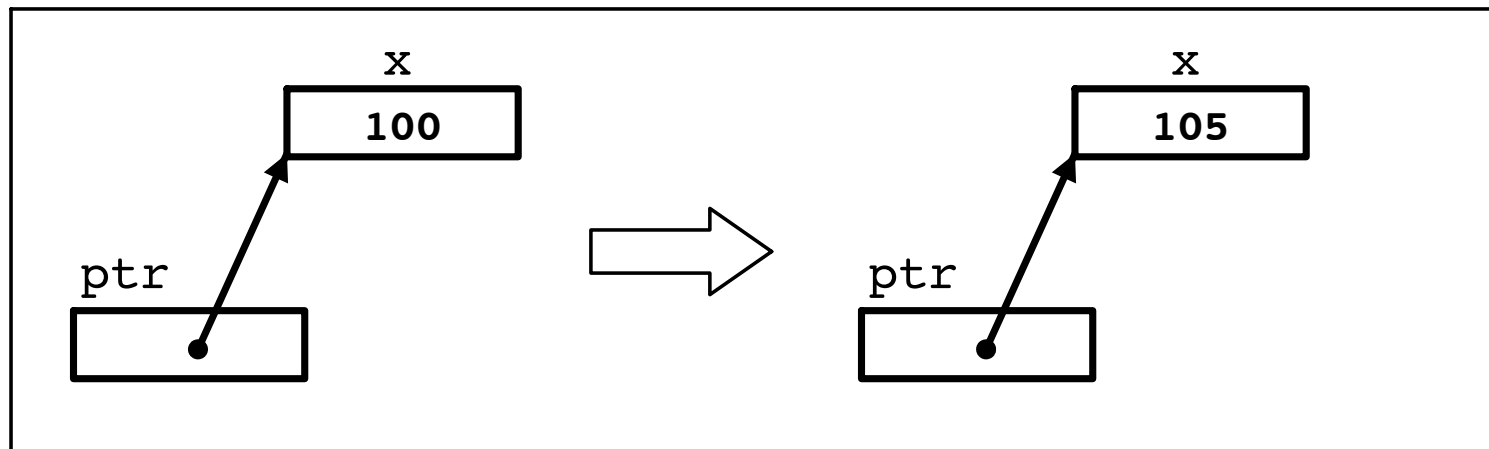
```
int x;
```

```
x = 100;
```

```
ptr = &x;
```

```
*ptr = *ptr + 5
```

เปลี่ยนค่าของ X
เป็น 105



โปรแกรมที่ 8.1

```
#include<stdio.h>
int main()
{
    int* pt;
    int a;
    a = 5;
    pt = &a;
    printf(" *pt = %d\n", *pt);
    *pt = 12;
    printf("a = %d\n", a);
    return 0;
}
```

<pre>*pt = 5 a = 12</pre>

โปรแกรมที่ 8.2

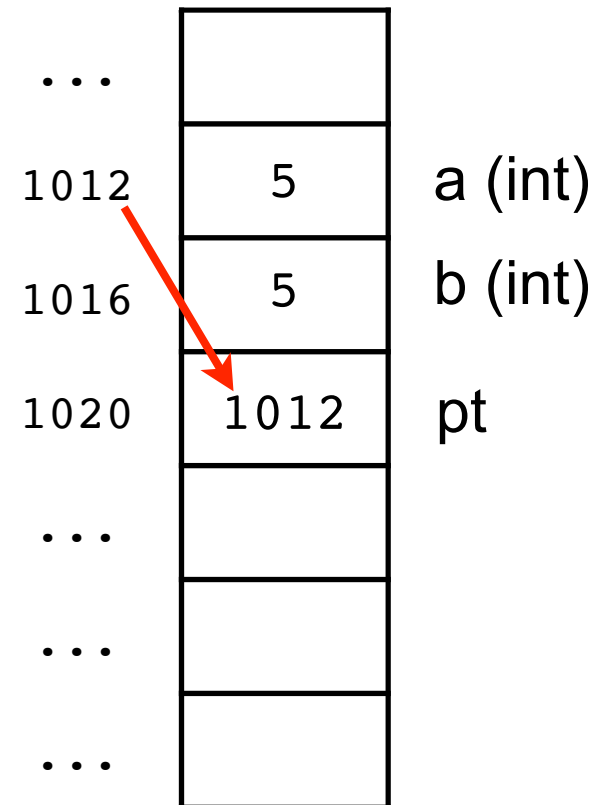
```
1  #include<stdio.h>
2  int main()
3  {
4      int a;
5      int b;
6      int* pt;
7      a = 5;
8      pt = &a;
9      b = *pt;
10     printf("a = %d\n",a);
11     printf("&a = %p\n",&a);
12     printf("b = %d\n",b);
13     printf("&b = %p\n",&b);
14     printf("pt = %p\n",pt);
15     printf("&pt = %p\n",&pt);
16     printf("*pt = %d\n",*pt);
17     return 0;
}
```

```
a = 5
&a = 0xbffff9ac
b = 5
&b = 0xbffff9a8
pt = 0xbffff9ac
&pt = 0xbffff9a4
*pt = 5
```

โปรแกรมที่ 8.2

```
1  #include<stdio.h>
2  int main()
3  {
4      int a;
5      int b;
6      int* pt;
7      a = 5;
8      pt = &a;
9      b = *pt;
10     printf("a = %d\n",a);
11     printf("&a = %p\n",&a);
12     printf("b = %d\n",b);
13     printf("&b = %p\n",&b);
14     printf("pt = %p\n",pt);
15     printf("&pt = %p\n",&pt);
16     printf("*pt = %d\n",*pt);
17     return 0;
}
```

```
a = 5
&a = 1012
b = 5
&b = 1016
pt = 1012
&pt= 1020
*pt= 5
```



โปรแกรมที่ 8.3

```
1  #include<stdio.h>
2  int main()
3  {
4      int a;
5      int b;
6      int *p;
7      int *q;
9      a = 5;
10     b = 10;
11     p = &a;
12     q = &b;
14     printf("a = %d\n",a);
15     printf("&a = %p\n",&a);
16     printf("b = %d\n",b);
17     printf("&b = %p\n",&b);
18     printf("p = %p\n",p);
19     printf("q = %p\n",q);
21     p = q;
22     *p = b + a;
23     printf("a = %d\n",a);
24     printf("&a = %p\n",&a);
25     printf("b = %d\n",b);
26     printf("&b = %p\n",&b);
27     printf("p = %p\n",p);
28     printf("q = %p\n",q);
29     return 0;
30 }
```

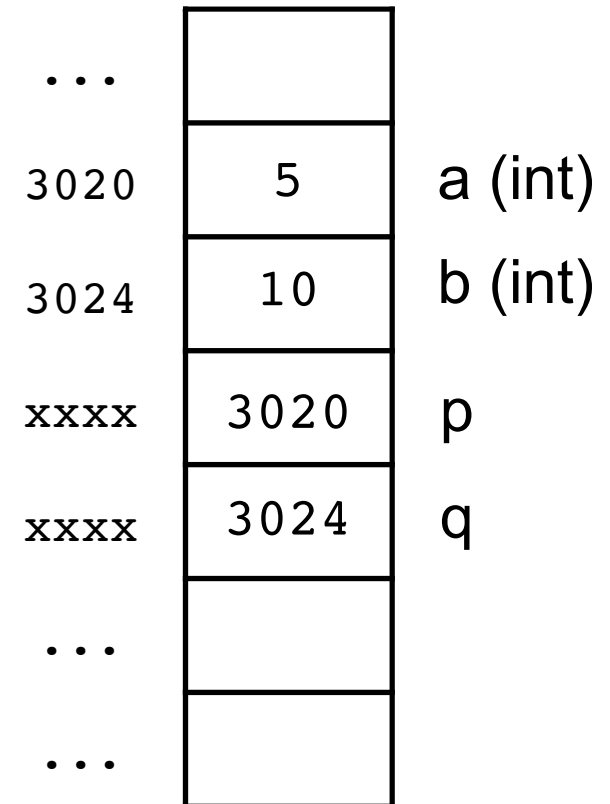
```
a = 5
&a = 3020
b = 10
&b = 3024
p = 3020
q = 3024
a = 5
&a = 3020
b = 15
&b = 3024
p = 3024
q = 3024
```

โปรแกรมที่ 8.3

```
1  #include<stdio.h>
2  int main()
3  {
4      int a;
5      int b;
6      int *p;
7      int *q;
9 → a = 5;
10 → b = 10;
11 → p = &a;
12 → q = &b;
14  printf("a = %d\n",a);
15  printf("&a = %p\n",&a);
16  printf("b = %d\n",b);
17  printf("&b = %p\n",&b);
18  printf("p = %p\n",p);
19  printf("q = %p\n",q);
21  p = q;
22  *p = b + a;
23  printf("a = %d\n",a);
24  printf("&a = %p\n",&a);
25  printf("b = %d\n",b);
26  printf("&b = %p\n",&b);
27  printf("p = %p\n",p);
28  printf("q = %p\n",q);
29  return 0;
30 }
```

```
a = 5
&a = 3020
b = 10
&b = 3024
p = 3020
q = 3024
```

```
a = 5
&a = 3020
b = 15
&b = 3024
p = 3024
q = 3024
```

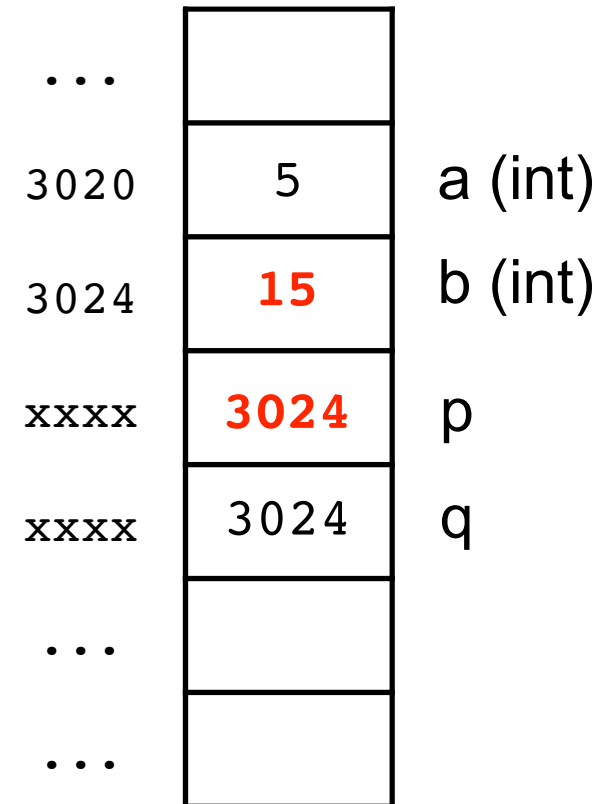


โปรแกรมที่ 8.3

```
1  #include<stdio.h>
2  int main()
3  {
4      int a;
5      int b;
6      int *p;
7      int *q;
9 → a = 5;
10 → b = 10;
11 → p = &a;
12 → q = &b;
14  printf("a = %d\n",a);
15  printf("&a = %p\n",&a);
16  printf("b = %d\n",b);
17  printf("&b = %p\n",&b);
18  printf("p = %p\n",p);
19  printf("q = %p\n",q);
21 → p = q;
22 → *p = b + a;
23  printf("a = %d\n",a);
24  printf("&a = %p\n",&a);
25  printf("b = %d\n",b);
26  printf("&b = %p\n",&b);
27  printf("p = %p\n",p);
28  printf("q = %p\n",q);
29  return 0;
30 }
```

```
a = 5
&a = 3020
b = 10
&b = 3024
p = 3020
q = 3024
```

```
a = 5
&a = 3020
b = 15
&b = 3024
p = 3024
q = 3024
```



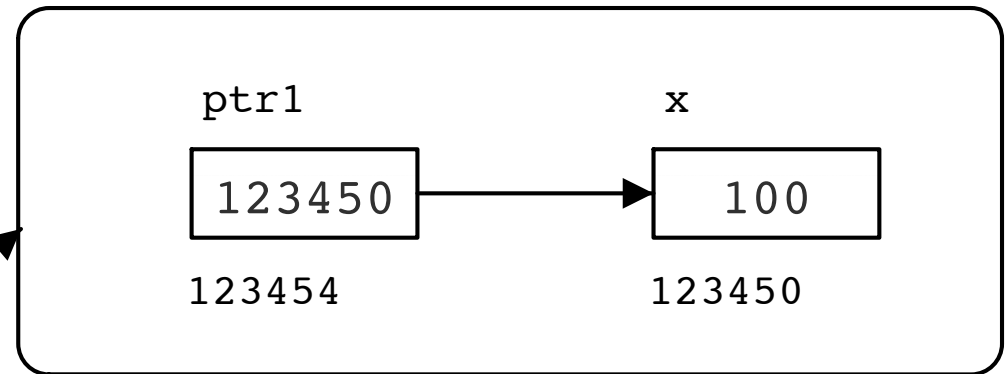
พอยน์เตอร์ชี้ไปที่พอยน์เตอร์

- การประกาศตัวแปรพอยน์เตอร์ชี้ไปที่พอยน์เตอร์ ใช้เครื่องหมาย ** หน้าชื่อตัวแปร

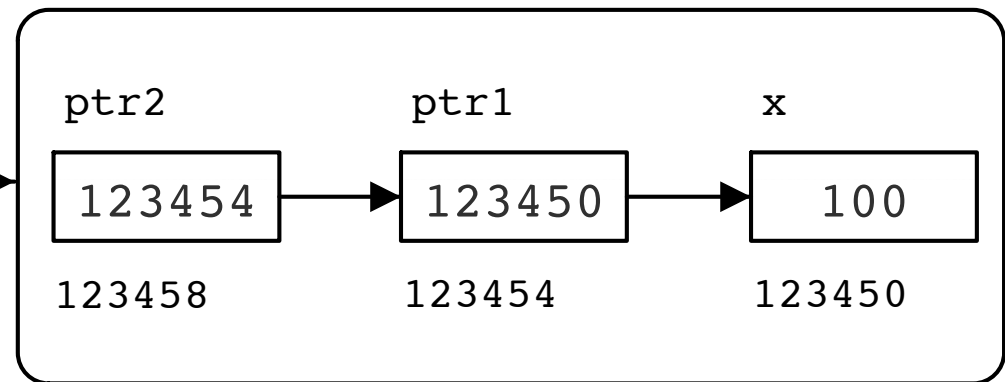
```
type** name;
```

พอยน์เตอร์ชี้ไปที่พอยน์เตอร์

ptr1 = &x



ptr2 = &ptr1



```
int x = 100;
int* ptr1;
int** ptr2;
ptr1 = &x;
ptr2 = &ptr1;
```

พอยน์เตอร์ชี้ไปที่พอยน์เตอร์

```
printf("%d\n", x);  
printf("%d\n", *ptr1);  
printf("%d\n", **ptr2);
```

ผลลัพธ์

100

100

100

พอยน์เตอร์กับอาร์เรย์

```
#include <stdio.h>

int main()
{
    int x[5]={1,2,3,4,5};
    printf("%p \n", x);
    printf("%d %d\n", *x, x[0]);
    return 0;
}
```

ผลลัพธ์

0xbffff838
1 1

อ้างชื่ออาร์เรย์คือ
การอ้างตำแหน่งที่อยู่
ตำแหน่งแรก

x →	1	x[0]
	2	x[1]
	3	x[2]
	4	x[3]
	5	x[4]

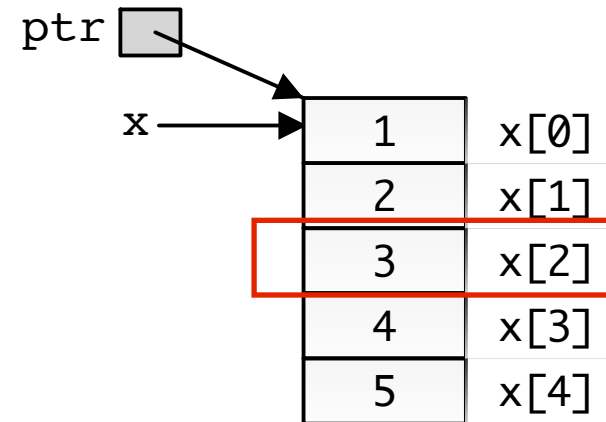
พอยน์เตอร์กับอาร์เรย์

```
#include <stdio.h>

int main()
{
    int x[5] = {1,2,3,4,5};
    int* ptr;
    ptr = x;
    printf("%p \n", ptr);
    printf("%d %d\n", *ptr, x[0]);
    return 0;
}
```

ผลลัพธ์

0xbffff838
1 1



`ptr = x` เทียบเท่ากับ `ptr = &x[0];`

`&x[2]` เทียบเท่ากับ `ptr + 2`

`x[2]` เทียบเท่ากับ `*(ptr + 2)` หรือ `ptr[2]`

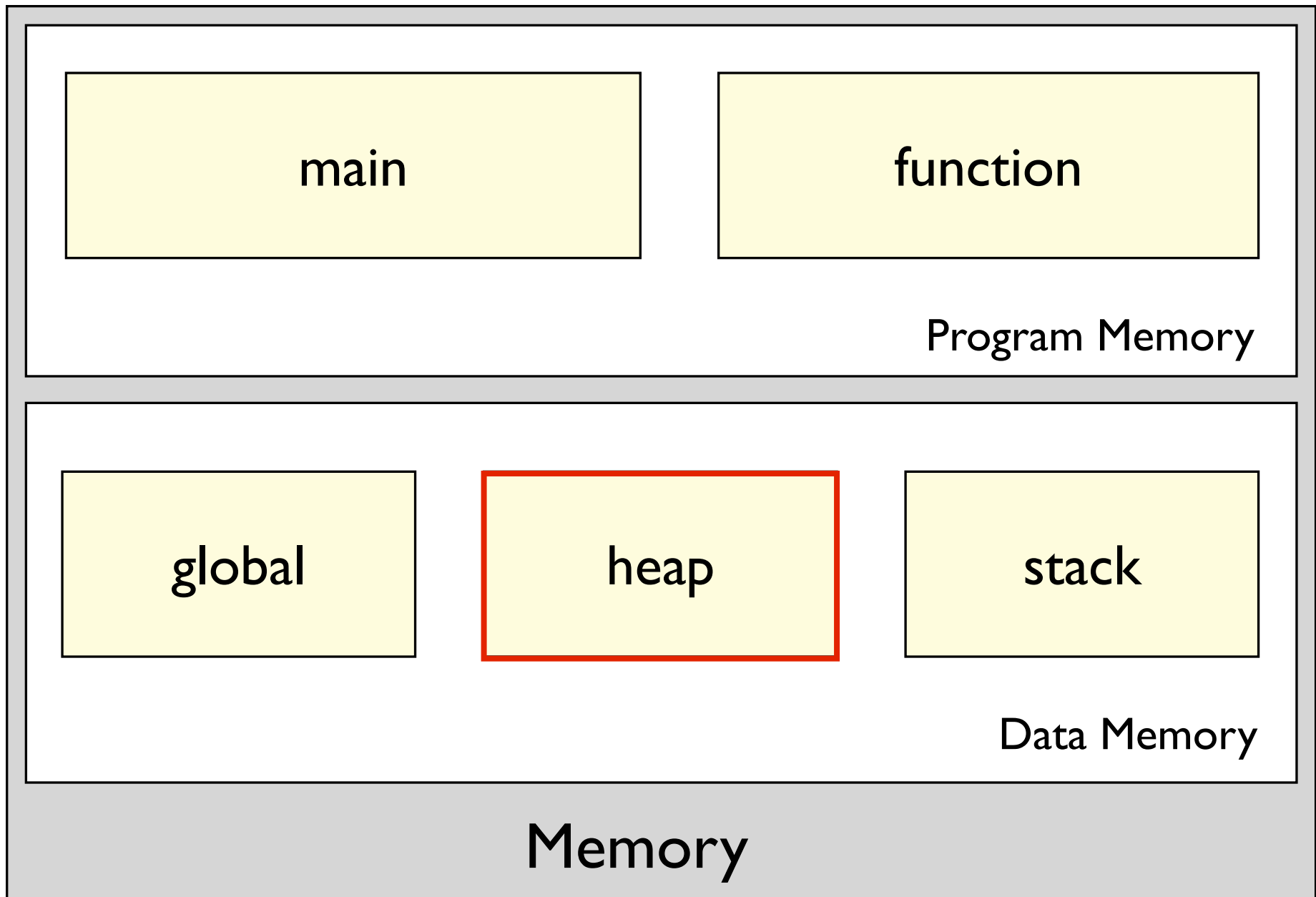
Dynamic memory allocation

- Dynamic memory allocation คือการใช้ข้อมูลที่สามารถระบุขนาดหรือขอเนื้อที่ในหน่วยความจำได้ในขณะที่โปรแกรมทำงาน (run-time)

Dynamic memory allocation

- Flexibility
- ไม่จำเป็นต้องกำหนดขนาดสูงสุดไว้ล่วงหน้า
- ประหยัดเนื้อที่ในหน่วยความจำ

การจัดการหน่วยความจำในภาษา C



การใช้งาน Dynamic Array

- การใช้งาน dynamic array มีขั้นตอนสำคัญ 4 ขั้นตอน

1. การประกาศตัวแปร

2. การจองเนื้อที่ในหน่วยความจำ

3. การใช้งาน

4. การคืนเนื้อที่ให้กับระบบ

การใช้งาน Dynamic Array

1. การประกาศตัวแปร โดยใช้พอยน์เตอร์ชี้ไปที่อาร์เรย์

```
type* array_name;
```

2. จองเนื้อที่ในหน่วยความจำ โดยใช้ฟังก์ชัน malloc โดยต้องระบุ stdlib.h

```
array_name =  
(type*)malloc(size_of_array*sizeof(type));
```

การใช้งาน Dynamic Array

3.การใช้งาน dynamic array (ตัวอย่างในหน้าถัดไป)

4.การคืนเนื้อที่ในหน่วยความจำ โดยใช้ฟังก์ชัน free

```
free(array_name);
```

การใช้งาน Dynamic Array

```
int someArray[10]
```

```
int* someArray;
```

```
someArray = (int*)malloc(10*sizeof(int));
```

ตัวอย่างการใช้ dynamic array

โปรแกรมที่ 8.4

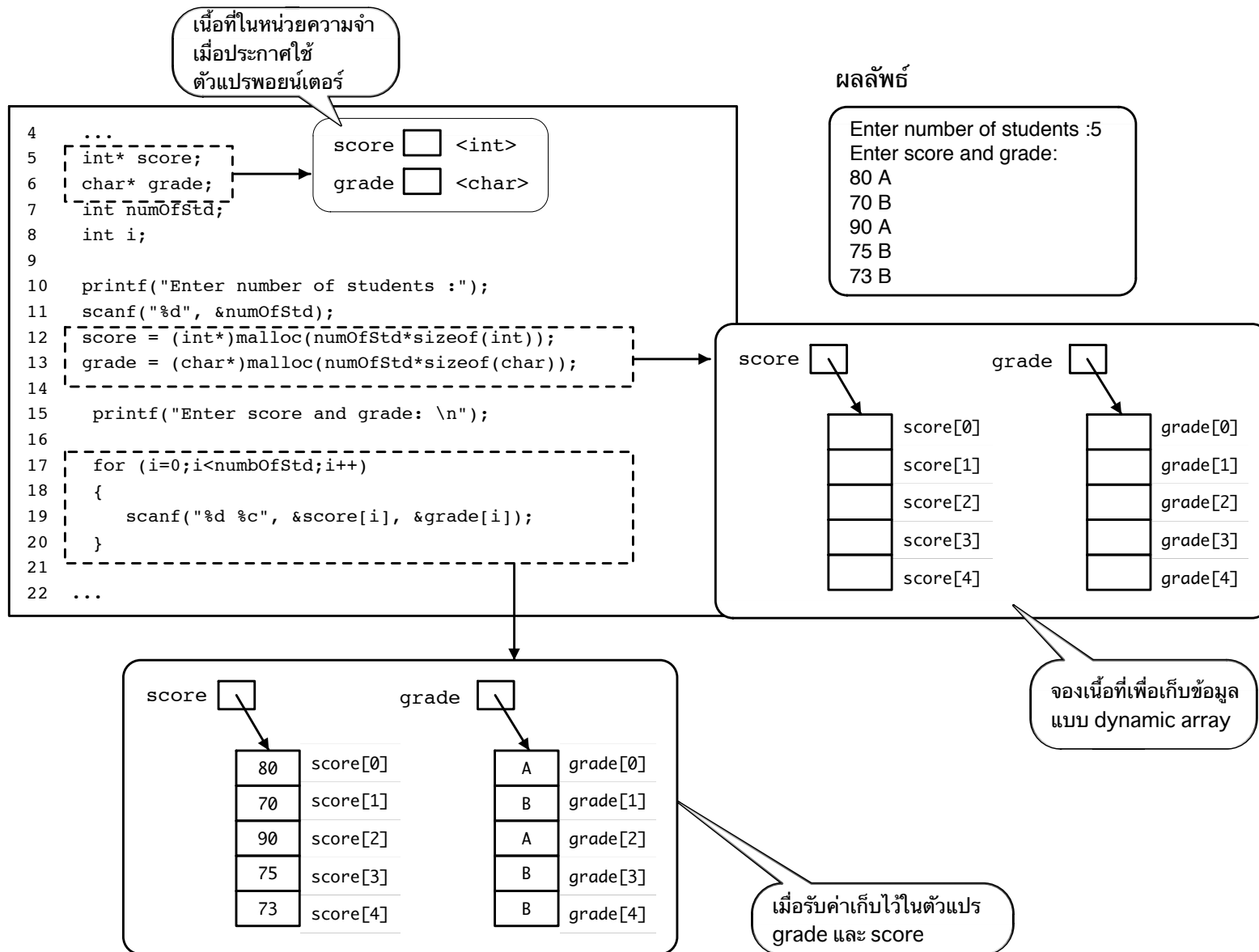
```
1 #include<stdio.h>
2 #include<stdlib.h>
3 int main()
4 {
5     int* score; // students' scores
6     char* grade; // students' grades
7     int numOfStd; // number of students
8     int i;
9
10    printf("Enter number of students :");
11    scanf("%d", &numOfStd);
12    score = (int*) malloc(numOfStd * sizeof(int));
13    grade = (char*) malloc(numOfStd * sizeof(char));
14
```

ตัวอย่างการใช้ dynamic array

โปรแกรมที่ 8.4

```
15     printf("Enter score and grade: \n");
16
17     for (i=0;i<numOfStd;i++)
18     {
19         scanf("%d %c", &score[i], &grade[i]);
20     }
21
22     for (i=0;i<numOfStd;i++)
23     {
24         printf("%d. %d %c \n", i+1, score[i], grade[i]);
25     }
26     free(score);
27     free(grade);
28     return 0;
29 }
```

การเก็บข้อมูลในหน่วยความจำของโปรแกรมที่ 8.4



แก้ไขโปรแกรมที่ 8.4

```
1  int main()
2  {
3      int numOfStd; // number of students
4      int i;
5
6      printf("Enter number of students :");
7      scanf("%d", &numOfStd);
8
9      int score[numOfStd]; // students' scores
10     char grade[numOfStd]; // students' grades
11
12     printf("Enter score and grade: \n");
13
14     for (i=0;i<numOfStd;i++)
15     {
16         scanf("%d %c", &score[i], &grade[i]);
17     }
18
19     for (i=0;i<numOfStd;i++)
20     {
21         printf("%d. %d %c \n", i+1, score[i], grade[i]);
22     }
23     return 0;
24 }
```

Array of Pointers

- Array of pointers ใช้เพื่อสร้าง dynamic array แบบ 2 มิติ
- ใช้เพื่อสร้าง ragged table หรือ ragged array ซึ่งหมายถึงตารางที่ไม่ทราบจำนวนที่แน่นอนของข้อมูล

Array of Pointers

- ตัวอย่าง ragged table

10	12	11			
1	22	21	12	32	
2	5	21	52		
41	32	15	1	12	15

การสร้าง ragged table

1. การประกาศตัวแปร โดยใช้พอยน์เตอร์ชี้ไปที่อาร์เรย์

```
int** arrPtr;
```

2. จองเนื้อที่ให้กับ array of pointers

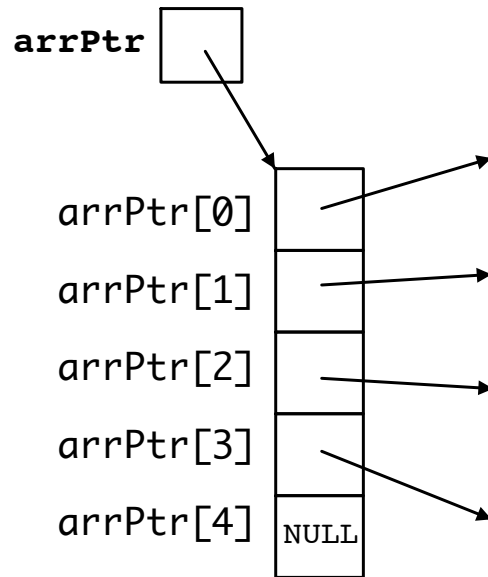
```
arrPtr = (int**)malloc(row_size*sizeof(int*));
```

การสร้าง ragged table

3. จองเนื้อที่ในแต่ละแถว เพื่อ เก็บข้อมูลชนิด `int`

```
arrPtr[row_index] =  
    (int*)malloc(column_size*sizeof(int));
```

ตัวอย่างการจองเนื้อที่ในหน่วยความจำ



```
arrPtr = (int**) malloc (5 * sizeof(int*));
```

```
arrPtr[0] = (int*) malloc (3 * sizeof(int));  
arrPtr[1] = (int*) malloc (5 * sizeof(int));  
arrPtr[2] = (int*) malloc (4 * sizeof(int));  
arrPtr[3] = (int*) malloc (6 * sizeof(int));  
arrPtr[4] = NULL;
```

จองเนื้อที่ตามจำนวนแถว
ของ array of pointers

จองเนื้อที่ตามจำนวน
คอลัมน์ของอาร์เรย์แต่ละตัว

กำหนดค่าของอาร์เรย์ตัว
สุดท้ายให้เป็น NULL

โปรแกรมที่ 8.5

```
3  int main()
4  {
5      int row;
6      int col;
7      int** arrPtr;
8      int rowArrNum;
9      int colArrNum;
10     printf("\nEnter the number of rows in the array: ");
11     scanf("%d", &rowArrNum);
12     arrPtr = (int**) malloc((rowArrNum + 1) * sizeof(int*));
13     for (row = 0; row < rowArrNum; row++) {
14         printf("\nEnter the number of columns in row %d: ", row + 1);
15         scanf("%d", &colArrNum);
16         arrPtr[row] = (int*) malloc((colArrNum + 1) * sizeof(int));
17         printf("\nEnter %d items: ", colArrNum);
18         arrPtr[row][0] = colArrNum;
19         for (col = 1; col <= colArrNum; col++)
20             scanf("%d", &arrPtr[row][col]);
21     }
22     arrPtr[row] = NULL;
23     row = 0;
24     printf("\n***Print items in the array***\n");
25     while (arrPtr[row]) {
26         printf("Row %d: \t", row + 1);
27         for (col = 1; col <= arrPtr[row][0]; col++)
28             printf("%d ", arrPtr[row][col]);
29         printf("\n");
30         row++;
31     }
32     return 0;
33 }
```

การเก็บข้อมูลในหน่วยความจำของโปรแกรมที่ 8.5

```
Enter the number of rows in the array: 3
Enter the number of columns in row 1: 3
Enter 3 items: 10 21 30
Enter the number of columns in row 2: 5
Enter 5 items: 22 53 61 27 18
Enter the number of columns in row 3: 4
Enter 4 items: 16 12 11 33
***Print items in the array***
Row 1:  10   21   30
Row 2:  22   53   61   27   18
Row 3:  16   12   11   33
```

ใส่จำนวนแถวเป็น 3

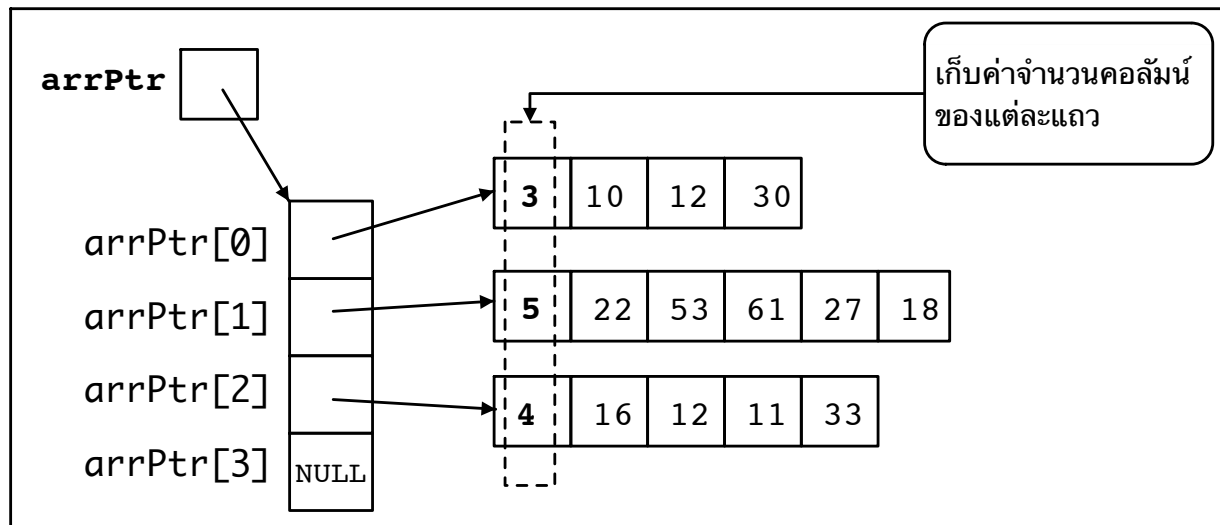
แถวแรกมี 3 คอลัมน์

ใส่จำนวนเต็ม 3 ค่า

แถวที่สองมี 5 คอลัมน์

ส่วนการแสดงค่า
ที่เก็บอยู่ใน arrPtr

a) ตัวอย่างผลลัพธ์เมื่อมีจำนวนแถว 3 แถว แต่ละแถวมีจำนวน 3, 5 และ 4 คอลัมน์



b) การเก็บข้อมูลในหน่วยความจำ